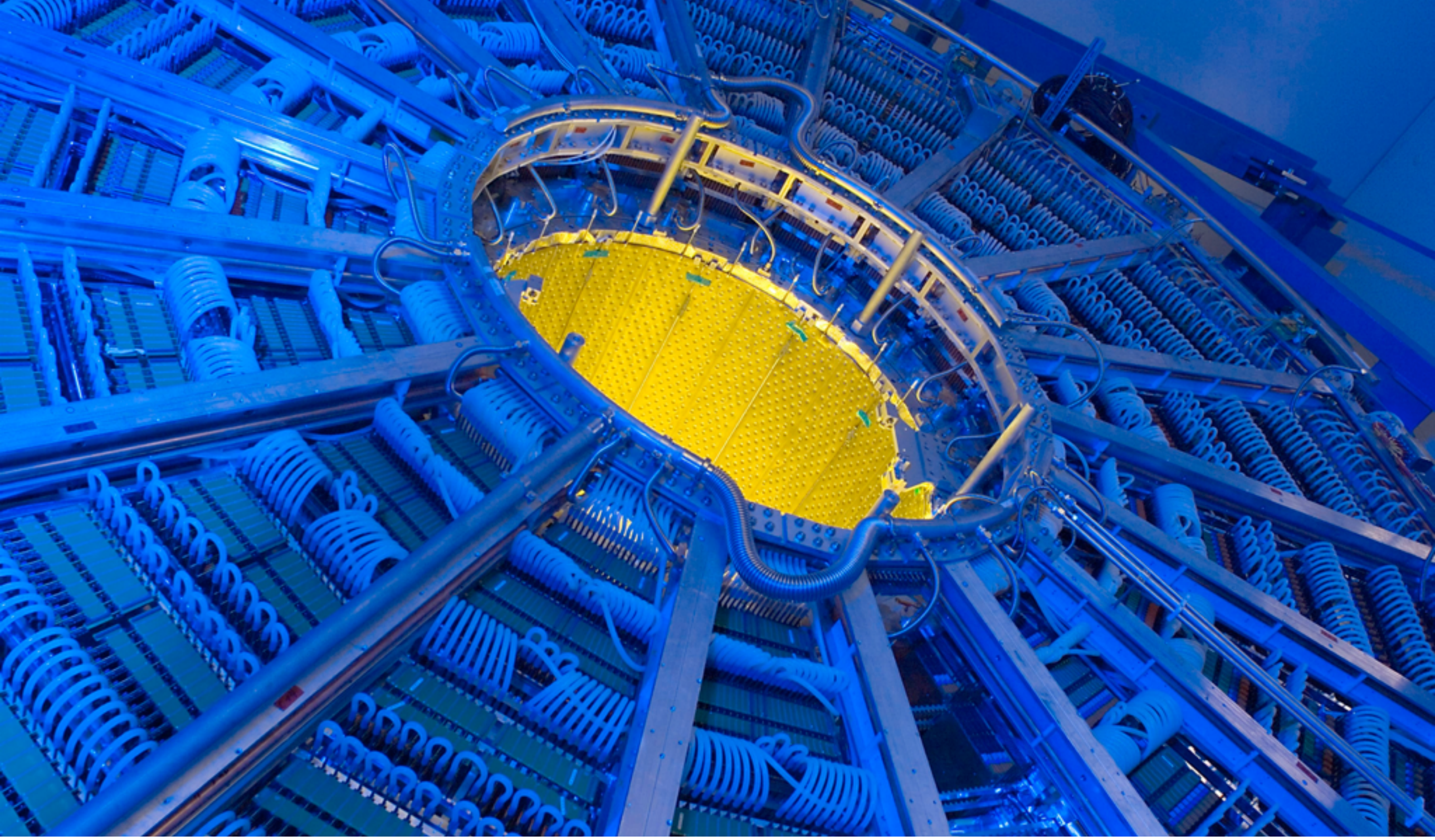


nanoAODs Status + AOD/ESD interface

A Festanti, M Floris,
JF Grosse-Oetringhaus,
R Russo, R Shahoyan, M Zimmerman

Part I: nanoAODs

Outline

- Use case:
  - The analyzer wants a **custom format** (to be analyzed with the analysis framework) containing only a very **limited subset** of the information
  - He defines the **few variables** he needs for tracks (which pass some **track cuts**) in events (which pass some) **event cuts**
  - An analysis task is run on the LEGO trains and produces the derived dataset (**nAOD**)
    - The nAOD is analyzed **locally** (if small enough) or on **CAF** or with the **trains**
    - When not needed any more, the nAOD is deleted: no centralized bookkeeping (**lifecycle** managed by PWGs)
      - Strict PWG disk quotas to be enforced

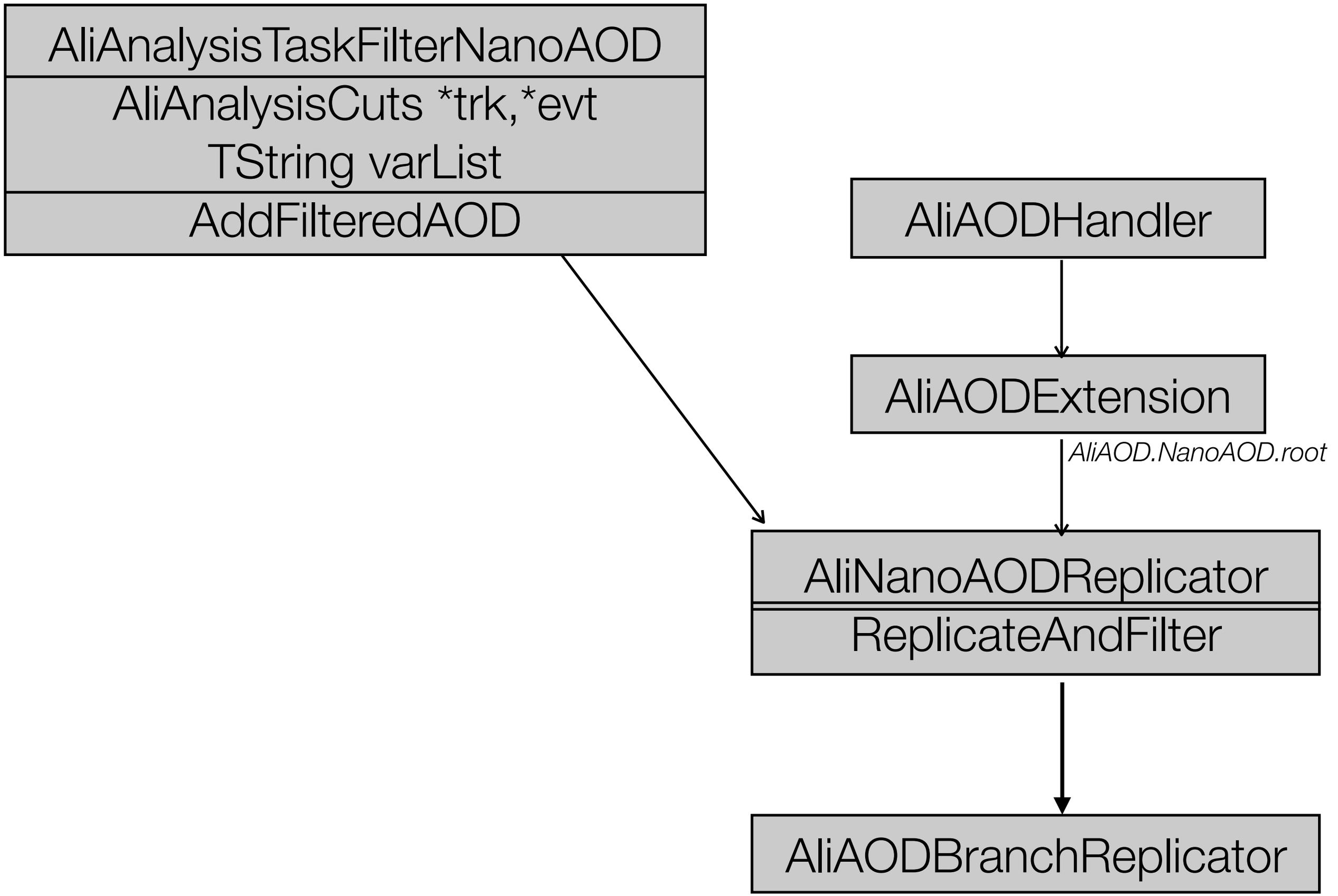
Some groups already use nanoAODs for specific cases (e.g. PWG-DQ, hyper-nuclei)
👍 More optimized / 👎 Need (at least) a custom replicator (= 2,3 additional classes)

Thanks for discussions, feedback and help:
Ramona, Stefania, Andreas, Peter, Andrei, Leonardo, Marco, Michael…

The big picture

AliAnalysisTaskFilterNanoAOD
AliAnalysisCuts *trk,*evt TString varList
AddFilteredAOD

The big picture



The big picture

AliAnalysisTaskFilterNanoAOD

AliAnalysisCuts *trk,*evt

TString varList

AddFilteredAOD

AliAODHandler

AliAODExtension

AliAOD.NanoAOD.root

AliNanoAODTrack

*

AliNanoAODHeader

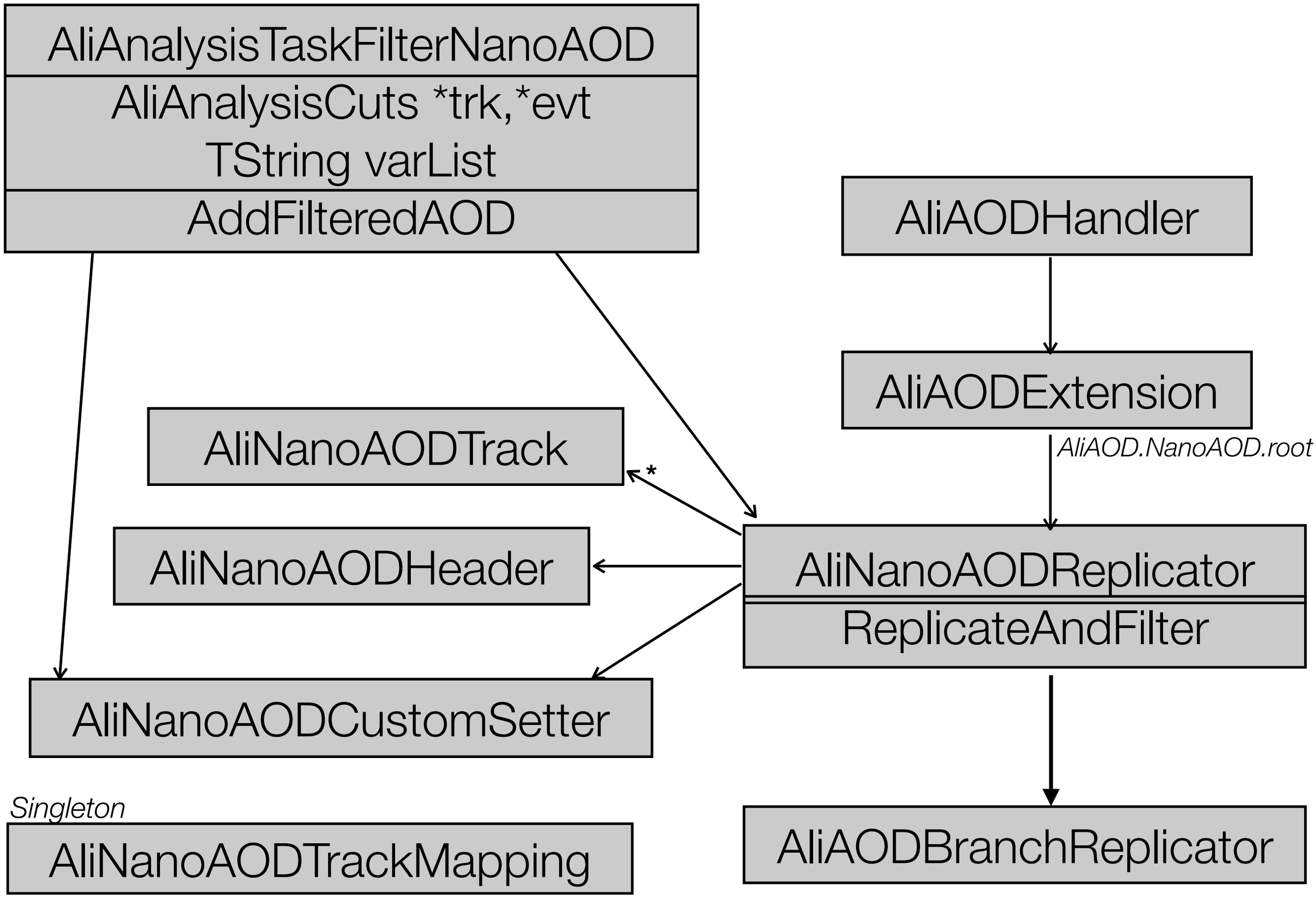
AliNanoAODReplicator

ReplicateAndFilter

AliNanoAODCustomSetter

AliAODBranchReplicator

The big picture



AliAODNanoTrack

Light weight track, for analysis-specific AOD
Only a subset of the info copied from AOD/ESD track

```
new AliNanoAODTrack(aodTrack, "pt,theta,phi,cstBayesProb")
```

Can be an
AOD or ESD track

Only those variables
are copied to the
special track

Custom Variables can be defined to store
derived quantities

Interface similar to AliAODTrack, derives from AliVTrack
(abundant use of AliFatals)

Summary (March 20, 2014)

- Fully **configurable nanoAOD** prototype developed
- Size: **10-30 times** smaller than a standard AOD (can try to further improve MC)
- For **I/O bound** analyses (spectra), similar gain in wall time (6-20 times faster)
  - Huge gain on the trains
  - Even more, Some analyses could be run locally on a powerful-enough desktop [the Spectra ESE analysis would require ~500 GB (data+MC)]
- For **CPU bound** analyses (correlations) gain is less obvious
  - ~33% faster
  - Can nevertheless help to reduce confusion (e.g. produce a hybrid tracks sample)
- It may not be the best solution for all analyses, but it can surely significantly improve some
  - We suggested to put it into production → Recommendation to continue

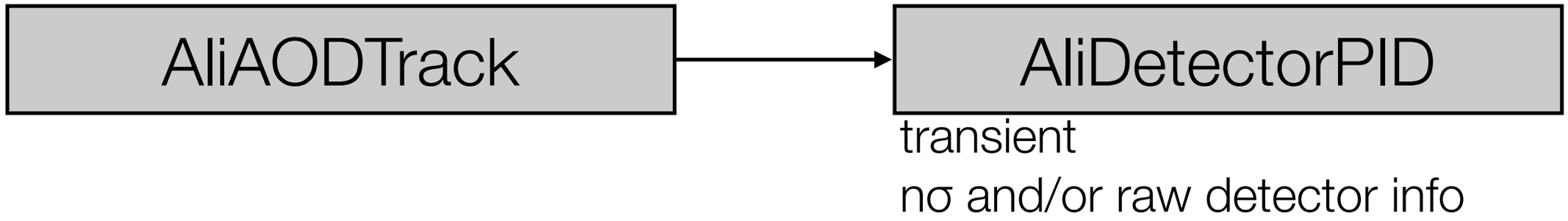
Changes to the analysis framework

- **Proposal** (March 2014):
  Change AliAOD{Track,Header}* to AliV{Track,Header}* in **AliAODEvent**
  - Would require patching some analysis tasks and framework classes
  - If a user can rely on the V track interface, no changes are needed in the task between processing AOD or nanoAOD format
  - If one needs specific methods, the tracks will have to be explicitly recasted.

- **Development in progress** (M. Zimmermann), in the NanoAODdev branch
  - Current tasks using AOD tracks / headers automatically patched to add a dynamic cast to AliAODtracks
  - Users will have to **check their own tasks** after the changes are committed!

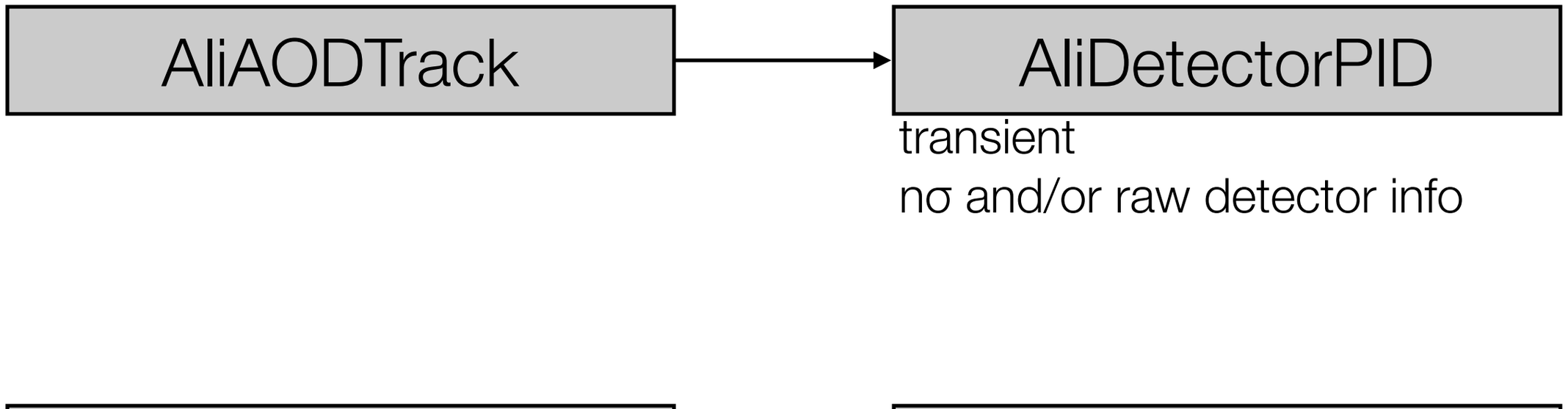
```
AliAODTrack *aodtrack =dynamic_cast<AliAODTrack*>(event->GetTrack(i));
if(!aodtrack) AliFatal("Not processing a standard AOD");
```

Integration with the PID framework



Thanks to Pietro and Jens!

Integration with the PID framework



Thanks to Pietro and Jens!

Integration with the PID framework

AliAODTrack → AliDetectorPID

transient

nσ and/or raw detector info

AliNanoAODTrack → AliDetectorPID

non-transient

filled by constructor, only with needed info (small)

nσ cached (fast)

bayesian should also work out of the box (but recomputed at run time)

(N.B. other "derived" AOD which use standard AOD objects, e.g. NucleExAODs, can use the standard PID framework)

Thanks to Pietro and Jens!

Part II:

Reconciling the AOD and ESD interface

Clean up of the AOD/ESD interface

- Why?
  - Homogeneous ESD/AOD interfaces (as much as possible)
  - Push interface back to the virtual classes
    - Benefit for nanoAOD vs AOD tasks
  - Clarify meaning if ambiguous members/getters (comments)
  - Remove duplications
- Requires changes to [Virtual,ESD,AOD][Events,Tracks,Vertices]
- Addressed the most obvious differences so far
- Will have to be reconsidered when we switch to a flat AOD structure

Ali[ESD|AOD]Events

- Implement: GetPrimaryVertexTPC, GetPrimaryVertexSPD (exist in ESD but not in AOD at the moment)
- Implement: GetVertex as an alias of GetPrimaryVertexSPD
- **Remove** AliAODEvent::GetNTracks (duplicates AliVEvent::GetNumberOfTracks)
- "AliAODTracklets ***AliAODEvent::GetTracklets**()" is equivalent to "AliMultiplicity ***AliESDEvent::GetMultiplicity**()", but:
  - AliAODTracklets and AliMultiplicity have a similar interface, but no common base class.
  - **Solution**: Common base class for AliMultiplicity and AliAODTracklets
  - **Rename** GetTracklets -> GetMultiplicity and move its definition to AliVEvent
- Remaining inconsistencies (list not complete)
  - Getters which return TClonesArrays only exist in AOD classes
  - Interface to detector information: FMD, PMD, HMPID, TZERO is different in ESD and AODs

Ali[AOD|ESD]Vertexes

- **SetX,Y,Z,Position** are (also) called **SetXv,Yv,Zv,XYZ** in the ESD (via AliVertex).
  - **Remove** the {Set,Get}Xv methods from AliVertex

- Implement IsFromVertexer3D(), IsFromVertexerZ() for AliAODVertex (In AliVVertex).

Ali[AOD|ESD]Tracks

- AOD: fType
  - Meaning was not clear: added additional comments
- AOD: XAtDCA, YAtDCA, ZAtDCA, PxAtDCA, PyAtDCA, PzAtDCA
  - Only make sense for constrained tracks (not available in ESDs): added explicit comments
- ESD interface: 2 ways of getting nclusters:
  - GetTRDNcls and GetTPCNcls + GetNcls(idet).
  - **Added** GetNcls(idet) also in AODs
- **Remove** AliAODTrack::GetFlags() (duplicates AliVTrack::GetStatus())
- **Remove** Bool_t AliAODTrack::IsTPCOnly()
  - obsolete bad naming (duplicates Bool_t IsTPCConstrained()

AliAODTrack (IsPrimaryCandidate)

- **Remove** the AODTrack::IsPrimaryCandidate
  - It is very misleading (returns true if the track passes any set of cuts). Owners of tasks contacted → no objections

```
- PWGCF/Correlations/DPhi/AliAnalysisTaskDiHadron.cxx
- PWGCF/FEMTOSCOPY/AliFemto/AliFemtoEventReaderAOD.cxx
- PWGGA/GammaConv/AliAnalysisTaskConversionQA.cxx
- PWGGA/GammaConv/AliAnalysisTaskMaterial.cxx
- PWGGA/GammaConv/AliAnalysisTaskResolution.cxx
- PWGHF/vertexingHF/AliAnalysisTaskSECharmFraction.cxx
- PWGHF/vertexingHF/AliAnalysisTaskSELambdac.cxx
- PWGJE/AliAnalysisTaskPartonDisc.cxx
```

AliAODTrack (GetChi2PerNDF)

AliAODTrack (GetChi2PerNDF)

- **Remove** AliAODTrack::GetChi2PerNDF() (barrel tracks)
  - Does not return chi2/NDF for barrel tracks (factor 2 missing), but ok for muons
  - Provides essentially the same information as the chi2/cluster variable we normally cut on; even numerically it is almost the same (as we typically require > 70 clusters)
  - There is no equivalent method in ESD; most ESD analyses simply cut on chi2/cluster
  - Name a bit misleading: uses TPC-only information, but not reflected in name
  - On the other hand: used (and useful!) for muons

AliAODTrack (GetChi2PerNDF)

- **Remove** AliAODTrack::GetChi2PerNDF() (barrel tracks)
  - Does not return chi2/NDF for barrel tracks (factor 2 missing), but ok for muons
  - Provides essentially the same information as the chi2/cluster variable we normally cut on; even numerically it is almost the same (as we typically require > 70 clusters)
  - There is no equivalent method in ESD; most ESD analyses simply cut on chi2/cluster
  - Name a bit misleading: uses TPC-only information, but not reflected in name
  - On the other hand: used (and useful!) for muons
- **Proposal**: remove AliAODTrack::GetChi2PerNDF and replace it by 2 methods:
  - AliAODTrack::GetTPCchi2
  - AliAODTrack::GetChi2PerNDF*[Muon?]*
    - If we keep the old name, it should AliFatal for non muon tracks
  - The 2 methods can share the same member variable and the backward compatibility can be handled via the schema evolution in the linked files

Backup

Tasks using GetChi2PerNDF

- PWG/FLOW/Tasks/AliAnalysisTaskFlowStrange.cxx
- PWG/FLOW/Tasks/AliAnalysisTaskPhiFlow.cxx
- PWG/FLOW/Tasks/AliFlowEventCuts.cxx
- PWG/muon/AliAnalysisMuonUtility.cxx
- PWG/muon/AliAnalysisTaskDimuonCFContainerBuilder.cxx
- PWG/muon/AliMuonInfoStoreRD.cxx
- PWGCF/Correlations/Base/AliAnalysisTaskCFTree.cxx
- PWGCF/Correlations/DPhi/AliAnalysisTaskDiHadron.cxx
- PWGCF/Correlations/DPhi/MuonHadron/AliAnalysisTaskDiMuonCorrelations.cxx
- PWGCF/Correlations/DPhi/PidPid/AliAnalysisTaskPidPidCorrelations.cxx
- PWGCF/Correlations/DPhi/TriggerPID/AliAnalysisTaskPIDCORR.cxx
- PWGCF/Correlations/DPhi/TriggerPID/AliTwoParticlePIDCorr.cxx
- PWGCF/EBYE/BalanceFunctions/AliAnalysisTaskAODFilterBitQA.cxx
- PWGCF/EBYE/BalanceFunctions/AliAnalysisTaskBF.cxx
- PWGCF/EBYE/BalanceFunctions/AliAnalysisTaskBFPsi.cxx
- PWGCF/EBYE/BalanceFunctions/AliAnalysisTaskEventMixingBF.cxx
- PWGCF/EBYE/Fluctuations/AliEbyEHigherMomentsEffContTask.cxx
- PWGCF/EBYE/Fluctuations/AliEbyEMultFluctuationTask.cxx
- PWGCF/EBYE/Fluctuations/AliHigherMomentsToyModel.cxx
- PWGCF/FEMTOSCOPY/AliFemto/AliFemtoEventReaderAOD.cxx
- PWGCF/FEMTOSCOPY/AliFemto/AliFemtoEventReaderStandard.cxx
- PWGCF/FEMTOSCOPY/Chaoticity/AliChaoticity.cxx
- PWGCF/FEMTOSCOPY/Chaoticity/AliFourPion.cxx
- PWGCF/FEMTOSCOPY/Chaoticity/AliThreePionRadii.cxx
- PWGCF/FLOW/Attic/AliAnalysisTwoParticleResonanceFlowTask.cxx
- PWGDQ/dielectron/AliDielectronVarManager.h
- PWGGA/CaloTasks/AliAnalysisTaskCaloFilter.cxx
- PWGGA/GammaConv/AliConversionTrackCuts.cxx
- PWGHF/hfe/AliAnalysisTaskFlowITSTPCTOFQCSP.cxx
- PWGHF/hfe/AliAnalysisTaskFlowTPCEMCalQCSP.cxx
- PWGHF/hfe/AliAnalysisTaskFlowTPCTOFEPSP.cxx
- PWGJE/AliPWG4HighPtTrackQA.cxx
- PWGJE/EMCALJetTasks/UserTasks/AliAnalysisTaskRhoVnModulation.cxx
- PWGLF/RESONANCES/AliRsnCutTrackQuality.cxx
- PWGLF/RESONANCES/AliRsnValueDaughter.cxx
- PWGLF/SPECTRA/ChargedHadrons/dNdPt/AlidNdPtAnalysisPbPbAOD.cxx
- PWGLF/SPECTRA/PiKaPr/TestAOD/AliAnalysisTaskSpectraBoth.cxx
- PWGLF/STRANGENESS/Hypernuclei/AliAnalysisTaskESDNuclExFilter.cxx
- PWGLF/STRANGENESS/Hypernuclei/AliAnalysisTaskReadNuclexAOD.cxx
- PWGLF/STRANGENESS/Hypernuclei/AliAODMCNuclExReplicator.cxx
- PWGLF/STRANGENESS/Hypernuclei/AliAODNuclExReplicator.cxx
- PWGUD/UPC/AliAnalysisTaskUpcK0sK0s.cxx
- PWGUD/UPC/AliAnalysisTaskUpcPsi2s.cxx

STEER/ESD/AliVertex.h

```
-  virtual Double_t GetXv() const { return fPosition[0]; }
-  virtual Double_t GetYv() const { return fPosition[1]; }
-  virtual Double_t GetZv() const { return fPosition[2]; }
+
   virtual Double_t GetX()  const { return fPosition[0]; }
   virtual Double_t GetY()  const { return fPosition[1]; }
   virtual Double_t GetZ()  const { return fPosition[2]; }
```

STEER/AOD/AliAODTrack.h

```
      enum AODTrk_t {kUndef = -1,
           kPrimary,
-          kSecondary,
-          kOrphan};
+          kFromDecayVtx,
+          kOrphan}; // Please note that this flag does not
guarantee that the particle is a Physical Primary, it simply
identifies the algorithm which was used to filter the track.
In general, the following associations are used (check the
filter macro to be sure, as this comment may be outdated):
+                                  //kPrimary: TPC only tracks,
global constrained tracks, primary tracks, kink mothers;
+                                  //kFromDecayVtx: bachelor tracks
from cascades, tracks from V0, kink daughters;
+                                  //kUndef:TRD matched tracks



  Char_t        fType;                  // Track Type, explanation close
to the enum AODTrk_t
```

STEER/AOD/AliAODTrack.h

```
  Double_t XAtDCA() const { return fPositionAtDCA[0]; } //makes sense
only for constrained tracks, returns dummy values for all other tracks
  Double_t YAtDCA() const { return fPositionAtDCA[1]; } //makes sense
only for constrained tracks, returns dummy values for all other tracks


  Double_t PxAtDCA() const { return fMomentumAtDCA[0]; } //makes sense
only for constrained tracks, returns dummy values for all other tracks
```